**Listing 1. Inicjalizacja modułu przetwornika DAC1**

void DAC1\_Load16bitInputData**(**uint16\_t input16BitData**)**

**{**

//DAC input reference range should be 16 bit

//Input data left justified.

DAC1CON0bits**.**DAC1FM **=** 1

//Loading 16bit data to DAC1

DAC1REFL **=** **(**uint8\_t**)** input16BitData**;**

DAC1REFH **=** **(**uint8\_t**)(**input16BitData **>>** 8**);**

//Loading DAC1 double buffer data to latch.

DAC1\_DoubleBufferLatch**();**

**}**

**Listing 2. Zapisanie 10-bitowych danych do przetwornika DAC1**

void DAC1\_Load10bitInputData**(**uint16\_t input10BitData**)**

**{**

//DAC input reference range should be 10bit.

//Input data right justified.

DAC1CON0bits**.**DAC1FM **=** 0**;**

//Loading 10bit data to DAC1

DAC1REFL **=** **(**uint8\_t**)** input10BitData**;**

DAC1REFH **=** **(**uint8\_t**)(**input10BitData **>>** 8**);**

//Loading DAC1 double buffer data to latch.

DAC1\_DoubleBufferLatch**();**

**}**

/\*\*

\* Loads data from DAC buffer onto the DAC output

\*/

#define DAC1\_DoubleBufferLatch() \

(DACLDbits.DAC1LD = 1)

**Listing 3. Pętla ustawiania napięcia wyjściowego zasilacza**

while**(**1**)**

**{**

**if(**frtime**==**1**)** //cykliczny pomiar napięcia i prądu

**{**

DispNap**(**PLV**,**1**);**

DispPr**(**PL**,**3**);**

Frame\_ms**(**500**);**

**}**

kod**=**GetEncoder**();**

**if(**kod**==**KOD\_IMP\_UP**)**

**{**

volt**=**volt**+**17.85**;**// zmiana o 0,5V - zwiększenie

**if(**volt**>**1000**)**

volt**=**1000**;**

DAC1\_Load10bitInputData**((**unsigned int**)**volt**);**//zapis do DAC

Delay\_ms**(**80**);**

DispNap**(**PLV**,**1**);**//zmierz i wyświetl napięcie

DispPr**(**PL**,**3**);**//zmierz i wyświetl prąd

Frame\_ms**(**1000**);**//kolejny pomiar za 1 sek

**}**

**if(**kod**==**KOD\_IMP\_DWN**)**

**{**

volt**=**volt**-**17.85**;**//zmiana o 0,5V - zmniejszenie

**if(**volt**<**35.7**)**

volt**=**35.7**;**

DAC1\_Load10bitInputData**((**unsigned int**)**volt**);**//zapis do DAC

Delay\_ms**(**80**);**

DispNap**(**PLV**,**1**);** //zmierz i wyświetl napięcie

DispPr**(**PL**,**3**);** //zmierz i wyświetl prąd

Frame\_ms**(**1000**);**//kolejny pomiar za 1 sek.

**}**

**if(**kod**==**KOD\_IMP\_ST**)**//przyciśnięcie ośki - ustawienie ograniczenia prądowego

**{**

Delay\_ms**(**200**)** **;**

SetI**();**//ustawienie progu zabezpieczenia prądowego

**}**

**}**

**Listing 4. Inicjalizacja modułu przetwornika A/C**

void ADC\_Initialize**(**void**)**

**{**

// set the ADC to the options selected in the User Interface

// ADGO stop; ADON enabled; CHS AN4;

ADCON0 **=** 0x11**;**

// ADFM right; ADNREF VSS; ADPREF FVR; ADCS Frc;

ADCON1 **=** 0xF3**;**

// TRIGSEL no\_auto\_trigger;

ADCON2 **=** 0x00**;**

// ADRESL 0;

ADRESL **=** 0x00**;**

// ADRESH 0;

ADRESH **=** 0x00**;**

**}**

**Listing 5. Konwersja analogowo-cyfrowa**

adc\_result\_t ADC\_GetConversion**(**adc\_channel\_t channel**)**

**{**

// wybór kanału pomiarowego

ADCON0bits**.**CHS **=** channel**;**

// włączenie modułu

ADCON0bits**.**ADON **=** 1**;**

// opóźnienie czasu akwizycji

\_\_delay\_us**(**ACQ\_US\_DELAY**);**

// Start konwersji

ADCON0bits**.**ADGO **=** 1**;**

// czekanie na zakończenie konwersji

**while** **(**ADCON0bits**.**ADGO**)**

**{**

**}**

// zwracanie wyniku pomiaru

**return** **((**ADRESH **<<** 8**)** **+** ADRESL**);**

**}**

**Listing 6. Pomiar napięcia i przeliczenie na napięcie w woltach**

void DispNap**(**unsigned char x**,** unsigned char y**)**

**{**

unsigned char i**;**

double vol**;**

vol**=**0**;**

**for(**i**=**0**;**i**<**10**;**i**++** **)**//wykonanie kolejnych 10 pomiarów

**{**

vol**=**vol**+**ADC\_GetConversion**(**4**);**//konwersja w kanale analogowym AN\_4

Delay\_ms**(**2**);**

**}**

vol**=**vol**/**10**;**//średnia z 10 pomiarów

vol**=**vol**\***2.8**;** //przeliczenie wartości odczytanej z ADC na wolty

vol**=**vol**/**100**;**

DispVolt**(**vol**,**x**,**y**);** //wyświetlenie pomiaru w woltach

volp**=**vol**;}**

**}**

**Listing 7. Wyświetlenie napięcia na wyświetlaczu**

void DispVolt**(**double vol**,**unsigned char x**,** unsigned char y **)**

**{**

char buf**[**10**];**

**if(**vol**<**10**)** //dla napięcia poniżej 10V

sprintf **(**buf**,**"%1.1fV "**,**vol**);**

**else** sprintf **(**buf**,**"%2.1fV "**,**vol**);**

PosLcd**(**x**,**y**);**

DispLcdRam**(**buf**);**

**}**

**Listing 8. Pomiar prądu zasilacza**

void DispPr**(**unsigned char x**,** unsigned char y**)**

**{**

unsigned char i**;**

double pr**;**

pr**=**0**;**

**for(**i**=**0**;**i**<**10**;**i**++)**

**{**

pr**=**pr**+**ADC\_GetConversion**(**5**);**

Delay\_ms**(**2**);**

**}**

pr**=**pr**/**10**;**

pr**=**pr**\***3**;**

pr**=**pr**/**1000**;**

DispI**(**pr**,**x**,**y**);**

**}**

void DispI**(**double pr**,**unsigned char x**,** unsigned char y **)**

**{**

char buf**[**10**];**

**if(**pr**<**1**)**

sprintf **(**buf**,**"%1.2fA "**,**pr**);**

**else**

sprintf **(**buf**,**"%2.1fA "**,**pr**);**

PosLcd**(**x**,**y**);**

DispLcdRam**(**buf**);**

**}**

**Listing 9. Ustawianie progu zabezpieczenia prądowego**

void SetI**(**void**)**

**{**

unsigned char kod**;**

PosLcd**(**15**,**4**);**DispLcd**(**"<-Set"**);**

DispIogr**(**PL**,**4**);**

**while(**1**)**

**{**

kod**=**GetEncoder**();**

**if(**kod**==**KOD\_IMP\_UP**)**

**{**

iogr**=**iogr**+**30.8**;**

**if(**iogr**>**925**)**

iogr**=**925**;**

DAC2\_Load10bitInputData**((**unsigned int**)**iogr**);**

Delay\_ms**(**80**);**

DispIogr**(**PL**,**4**);**

Delay\_ms**(**80**);**

DispNap**(**PLV**,**1**);**

DispPr**(**PL**,**3**);**

**}**

**if(**kod**==**KOD\_IMP\_DWN**)**

**{**

iogr**=**iogr**-**30.8**;**

**if(**iogr**<**30.8**)**

iogr**=**30.8**;**

DAC2\_Load10bitInputData**((**unsigned int**)**iogr**);**

Delay\_ms**(**80**);**

DispIogr**(**PL**,**4**);**

Delay\_ms**(**80**);**

DispNap**(**PLV**,**1**);**

DispPr**(**PL**,**3**);**

**}**

**if(**kod**==**KOD\_IMP\_ST**)**

**{**

Delay\_ms**(**200**);**

PosLcd**(**15**,**4**);**DispLcd**(**" "**);**

DispIogr**(**PL**,**4**);**

**return;**

**}**

**}**

**}**

**Listing 10. Procedura odczytania słowa z pamięci Flash**

uint16\_t FLASH\_ReadWord**(**uint16\_t flashAddr**)**

**{**

uint8\_t GIEBitValue **=** INTCONbits**.**GIE**;** // zapisanie stanu bitu GIE

INTCONbits**.**GIE **=** 0**;** // Zablokowanie przerwań

PMADRL **=** **(**flashAddr **&** 0x00FF**);**

PMADRH **=** **((**flashAddr **&** 0xFF00**)** **>>** 8**);**

PMCON1bits**.**CFGS **=** 0**;** //

PMCON1bits**.**RD **=** 1**;** // zainicjowanie odczytu danych

NOP**();**

NOP**();**

INTCONbits**.**GIE **=** GIEBitValue**;** // Odtworzenie stanu bitu GIE

**return** **((**PMDATH **<<** 8**)** **|** PMDATL**);**

**}**

**Listing 11. Procedura zapisu wiersza do pamięci Flash**

int8\_t FLASH\_WriteBlock**(**uint16\_t writeAddr**,** uint16\_t **\***flashWordArray**)**

**{**

uint16\_t blockStartAddr**=(**uint16\_t**)(**writeAddr**&((**END\_FLASH**-**1**)^(**ERASE\_FLASH\_BLOCKSIZE**-**1**)));**

uint8\_t GIEBitValue**=**INTCONbits**.**GIE**;** // Save interrupt enable

uint8\_t i**;**

// zapis Flash musi się rozpoczynać od początku wiersza

**if(** writeAddr **!=** blockStartAddr **)** **return** **-**1**;**

// Zapisanie stanu zezwolenia na przerwania

INTCONbits**.**GIE **=** 0**;**

// sekwencja kasowania wiersza

FLASH\_EraseBlock**(**writeAddr**);**

// sekwencja zapisu wiersza

PMCON1bits**.**CFGS **=** 0**;** // Deselect Configuration space

PMCON1bits**.**WREN **=** 1**;** // Enable wrties

PMCON1bits**.**LWLO **=** 1**;** // Only load write latches

**for** **(**i**=**0**;** i**<**WRITE\_FLASH\_BLOCKSIZE**;** i**++)**

**{**

//8 młodszych bitów adresu

PMADRL **=** **(**writeAddr **&** 0xFF**);**

// 6 starszych bitów adresu

PMADRH **=** **((**writeAddr **&** 0xFF00**)** **>>** 8**);**

// zapisanie danych

PMDATL **=** flashWordArray**[**i**];**

PMDATH **=** **((**flashWordArray**[**i**]** **&** 0xFF00**)** **>>** 8**);**

**if(**i **==** **(**WRITE\_FLASH\_BLOCKSIZE**-**1**))**

**{**

// start zapisu pamięci Flash

PMCON1bits**.**LWLO **=** 0**;**

**}**

PMCON2 **=** 0x55**;**

PMCON2 **=** 0xAA**;**

PMCON1bits**.**WR **=** 1**;**

NOP**();**

NOP**();**

writeAddr**++;**

**}**

// zablokowanie zapisu

PMCON1bits**.**WREN **=** 0**;**

// odtworzenie stanu zezwolenie na przerwania

INTCONbits**.**GIE **=** GIEBitValue**;**

**return** 0**;**

**}**